Breviar laborator 13

Design Patterns

Prezentare generala

Un șablon de proiectare descrie o problema care se întâlnește în mod repetat în proiectarea programelor și soluția generala pentru problema respectiva, astfel încât sa poata fi utilizata oricând, dar nu în același mod de fiecare data. Soluția este exprimata folosind claseși obiecte. Atât descrierea problemei cât și a soluției sunt abstracte astfel încât sa poata fi folosite în multe situații diferite. Scopul șabloanelor de proiectare este de a asista rezolvarea unor probleme similare cu unele deja întâlniteși rezolvate anterior. Ele ajuta la crearea unui limbaj comun pentru comunicarea experienței despre aceste problem și soluțiile lor.

Cele 4 elemente cheie care definesc un șablon de proiectare sunt urmatoarele:

1. Numele șablonului de proiectare – având în vedere ca exista multe tipuri de șabloane de proiectare, este important ca fiecaruia sa îi fie aplicat un nume sugestiv, în strânsa legatura cu problema pe care o rezolva, care sa permita identificarea rapida a acestuia și a documentației aferente.

2. Descrierea situatiei în care poate fi aplicat – este foarte important sa știm care sunt situațiile în care putem aplica un șablon de proiectare. De aceea, este nevoie sa se prezinte o descriere a problemei și a contextului în care ar putea sa apara. Aceasta descriere poate fi realizata din perspective diferite:

• Ar putea sa fie o descriere axata pe aspecte specifice de proiectare, cum ar fi modul de reprezentare a diverșilor algoritmi folosind principiile programarii orientate pe obiecte.

• Poate fi realizata o descriere care sa conțina o ierarhie de clase sau o structura de obiecte care sunt implicate în implementarea șablonului de proiectare.

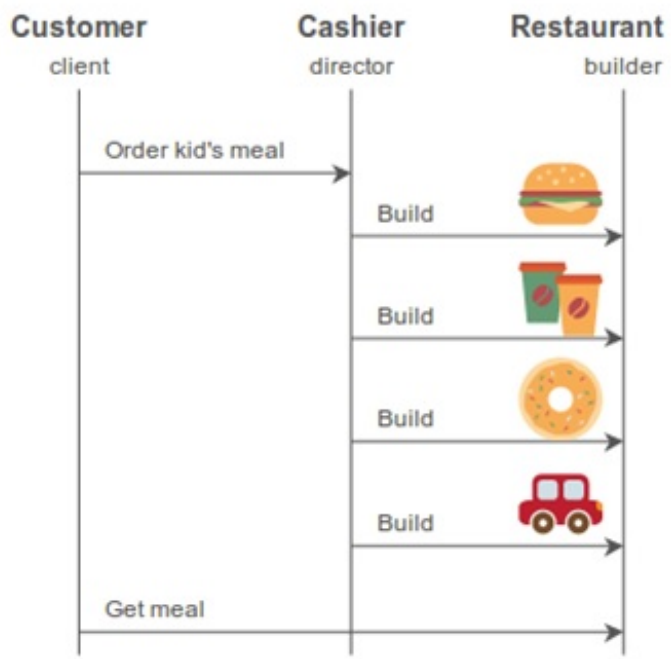
• Uneori, este important sa specificam o lista de condiții care trebuie sa fie îndeplinite pentru a putea aplica șablonul de proiectare. În acest caz, descrierea trebuie sa cont,ina aceasta lista de condiții.

3. Descrierea soluției – descrie elementele care alcatuiesc proiectarea, relațiile, responsabilitațile și colaborarile acestora. Este indicat ca soluția sa nu conțina doar codul complet, ci și o descriere formala a unei problem și modul în care o interacțiune generala a conceptelor (clase și obiecte) poate rezolva problema.

4. Rezultatele și consecințele utilizarii – reutilizarea codului reprezinta, adesea, un factor esențial în programarea orientata pe obiecte, motiv pentru care pentru un șablon de proiectare trebuie sa fie prezentate consecințele pe care le au folosirea acestuia asupra flexibilităț ii, extensibilitații sau portabilitații soluției software.

Builder

Acest pattern este folosit în restaurantele de tip fast food care furnizeaza meniul pentru copii. Un meniu pentru copii consta de obicei într-un fel principal, unul secundar, o bautura si o jucarie. Pot exista variatii în ceea ce privește conținutul mediului, dar procesul de creare este acelasi. Fie ca la felul principal se alege un hamburger sau un cheesburger procesul va fi același. Vânzatorul le va indica celor din spate ce sa puna pentru fiecare fel de mâncare, pentru bautura si jucarie. Toate acestea vor fi puse într-o punga si servite clientilor.



Acest sablon realizeaza separarea constructiei de obiecte complexe de reprezentarea lor astfel încât acelasi proces sa poata crea diferite reprezentari. Builder-ul creeaza parti ale obiectului complex de fiecare data când este apelat si retine toate starile intermediare. Când departamentul este terminat, clientul primeste rezultatul de la builder. În acest mod, se obtine un control mai mare asupra procesului de constructie de noi obiecte. Spre deosebire de alte pattern-uri, din categoria creational, care creau produsele într-un singur pas, pattern-ul Builder construieste un produs pas cu pas la comanda coordonatorului. În cadrul acestei aplicatii, pattern-ul este folosit pentru a instantia departamentele. Pentru a întelege cum ar trebui folosit acest pattern, puteti urmari exemplul de mai jos.

```
public class User  {
    private final String firstName; // required
    private final String lastName; // required
    private final int age; // optional
    private final String phone; // optional
    private final String address; // optional
    private User(UserBuilder builder) {
        this.firstName = builder.firstName;
        this.lastName = builder.lastName;
        this.age = builder.age;
        this.phone = builder.phone;
        this.address = builder.address;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public int getAge() {
        return age;
    }
    public String getPhone() {
        return phone;
    }
    public String getAddress() {
        return address;
    }
    public String toString() {
        return "User:"+this.firstName+" "+this.lastName+" "+this.age+" "+this.phone+" " + this.address;
    }
    public static class UserBuilder {
        private final String firstName;
        private final String lastName;
        private int age;
```

```
        private String phone;
        private String address;
        public UserBuilder(String firstName, String lastName) {
            this.firstName = firstName;
            this.lastName = lastName;
        }
        public UserBuilder age(int age) {
            this.age = age;
            return this;
        }
        public UserBuilder phone(String phone) {
            this.phone = phone;
            return this;
        }

        public UserBuilder address(String address) {
            this.address = address;
            return this;
        }
        public User build() {
            return new User(this);
        }
    }

  public static void main(String[] args) {
        User user1 = new User.UserBuilder("Lokesh", "Gupta")
                              .age(30)
                              .phone("1234567")
                              .address("Fake address 1234")
                              .build();
        User user2 = new User.UserBuilder("Jack", "Reacher")
                              .age(40)
                              .phone("5655")
                              //no address
                              .build();
   }
}
```